/Projet Poker

- Karim **CHARLEUX**
- Guillaume **ARRIGONI**
- Loris **DRID**
- Yacine **MERIOUA**

/SOMMAIRE

/01
/Fonctionnalités

Les règles du poker :

/01

/Fonctionnalités réalisées

- ✅ Toutes les règles du jeu la main poker
- ✅ Deux joueurs peuvent écrire et confronter leurs mains
- ✅ Affichage des résultats de la partie

/Fonctionnalités non réalisées

- ❌ Pouvoir enchaîner les parties
- ❌ Fournir la bonne syntaxe lors d’une erreur du joueur pendant la saisie des cartes

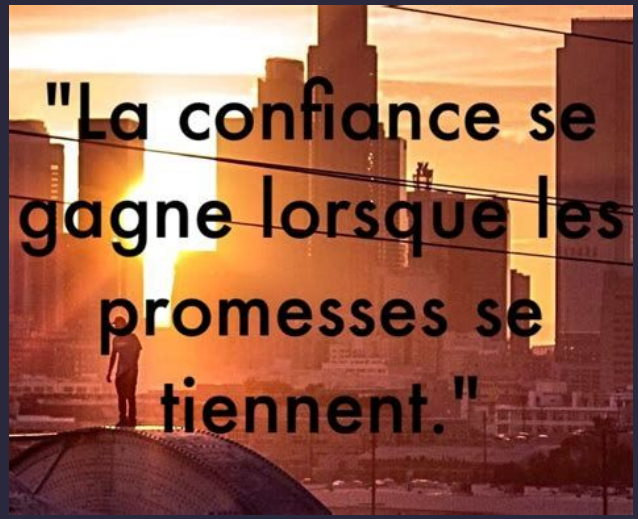
/01

/Démo

- Partie n°1 : Quinte Flush **VS** Suite
- Partie n°2 : Full **VS** Carré
- Partie n°3 : Brelan **VS** Pair
- Partie n°4 : Double Pair **VS** Double Pair

/02
/Confiance

+Test



/02

/Parties en confiance

- La classe Comparison.java
- La base du jeu (Color & Value & Card)

/02

/Parties pas en confiances

- Manque des tests sur GamePoker
- Manque des tests sur certaines exceptions

/02

Exécution des Tests…

/03
/Qualité

+Test
pertinants

/03

/Qualité du code

- Le code est maintenable
- Le code est lisible, compréhensible
- Le code est commenté avec la JavaDoc

/03

/Tests paramétrés

```
private static Stream<Arguments> provideComparison(){
    return Stream.of(
            Arguments.of(comparisonBrelanE, Combination.BRELAN),
            Arguments.of(comparisonCarreE, Combination.CARRE),
            Arguments.of(comparisonFullE, Combination.FULL),
            Arguments.of(comparisonFlushE, Combination.FLUSH),
            Arguments.of(comparisonDoublePairE, Combination.TWO_PAIR),
            Arguments.of(comparisonPairE, Combination.PAIR),
            Arguments.of(comparisonHigherCardE, Combination.HIGHCARD),
            Arguments.of(comparisonStraightE, Combination.SUITE),
            Arguments.of(comparisonStraightFlushE, Combination.STRAIGHT_FLUSH),
            Arguments.of(comparisonStraightFlushW, Combination.STRAIGHT_FLUSH),
            Arguments.of(comparisonCarreW, Combination.CARRE),
            Arguments.of(comparisonFullW, Combination.FULL),
            Arguments.of(comparisonFlushW, Combination.FLUSH),
            Arguments.of(comparisonStraightW, Combination.SUITE),
            Arguments.of(comparisonBrelanW, Combination.BRELAN),
            Arguments.of(comparisonDoublePairW, Combination.TWO_PAIR),
            Arguments.of(comparisonPairW, Combination.PAIR)
            );
}
2 usages  Loris
private static Stream<Arguments> provideComparisonEquals(){
    return Stream.of(
            Arguments.of(comparisonSameFlush, Combination.EQUALITY),
            Arguments.of(comparisonSameDoublePair, Combination.EQUALITY),
            Arguments.of(comparisonSamePair, Combination.EQUALITY),
            Arguments.of(comparisonSameHigherCard, Combination.EQUALITY),
            Arguments.of(comparisonSameStraightFlush, Combination.EQUALITY),
            Arguments.of(comparisonSameStraight, Combination.EQUALITY)
    );
```

/03

/Tests paramétrés

```
@ParameterizedTest
@MethodSource("provideComparison")
void testIfTheComparisonWinningValueIsTrue(Comparison comparison) { assertTrue(comparison.getWinning().get()); }

 Loris +1
@ParameterizedTest
@MethodSource({"provideComparisonEquals"})
void testIfWinningValueIsEmptyWhenEquality(Comparison comparison) { assertTrue(comparison.getWinning().isEmpty()); }

 Loris
@ParameterizedTest
@MethodSource({"provideComparison", "provideComparisonEquals"})
void testIfWinningCombinationCorrespondsToTheWinnerCombination(Comparison comparison, Combination combinationWinner) {
    assertEquals(combinationWinner, comparison.getWinningCombination());
}
```

/04
/Qui a fait quoi

/04

/En groupe

- Les classes suivantes
  - GamePoker
  - Card
  - Value
  - Comparison

/04

- Guillaume
  - Première version de la classe Comparison
  - La classe HandPoker
- Karim
  - Superviser un peu tout le monde
  - La classe Combination
- Loris
  - La classe Color
  - Les tests Comparison
- Yacine
  - Test HandPoker
  - Test Card

/Plus individuel

/MERCI !

/Vous avez des questions ?